Feedback-based Access Schemes in CR Networks: A Reinforcement Learning Approach

Ehab M. El-Guindy[‡], Karim G. Seddik[‡], Amr A. El-Sherif[+†], and Tamer ElBatt[⊥]

[‡]Electronics and Communications Engineering Department, American University in Cairo, New Cairo 11835, Egypt.
[+]Wireless Intelligent Networks Center (WINC), Nile University, Giza 12588, Egypt.
[†]Department of Electrical Engineering, Alexandria University, Alexandria 21544, Egypt.
[⊥]Computer Science and Engineering Department, American University in Cairo, New Cairo 11835, Egypt.
Email: eelguindy@aucegypt.edu, kseddik@aucegypt.edu, aelsherif@nu.edu.eg, tamer.elbatt@aucegypt.edu

***Abstract*—In this paper, we propose a Reinforcement Learning-based MAC layer protocol for cognitive radio networks, based on exploiting the feedback of the Primary User (PU). Our proposed model relies on two pillars, namely an infinite-state Partially Observable Markov Decision Process (POMDP) to model the system dynamics besides a queuing-theoretic model for the PU queue, where the states represent whether a packet is delivered or not from the PU's queue and the PU channel state. Based on the stability constraint for the primary user queue, the quality of service (QoS) for the PU is guaranteed. Towards the paper's objectives, three Reinforcement Learning approaches are studied, namely Q-Learning, Deep Q-Network (DQN), and Deep Deterministic Policy Gradient (DDPG). Our ultimate objective is to enhance the channel access techniques in the MAC protocols by solving the POMDP without any prior knowledge of the environment.**

***Index terms*—** Cognitive Radio, Reinforcement learning, Queue Stability

I. Introduction

The evolution of cognitive Radios by authorizing Secondary Users (SUs) to exploit the underutilized spectrum of Primary Users (PUs) is essential to solving the scarcity problem of the unused frequencies. However, SUs access is constrained by not affecting the PUs network. Quality of service (QoS) level is always guaranteed for the PUs in the presence of SUs. SUs medium access control (MAC) schemes have attracted a lot of interest over the last few years [1].

Many papers have focused on the use of the PU(s) feedback information to devise better SUs' MAC schemes. Eswaran *et al.* [2] has considered the use of automatic repeat request (ARQ) messages, that were received by the SU, to perceive the packet rate achieved by the PU; the aim is to maximize the secondary throughput while ensuring a minimal PU packet rate. Furthermore, based on the primary link feedback, secondary power control is explored in [3]. While maintaining particular PU QoS requirements, the goal was to maximize the SU utility in a distributed scheme. Moreover, in [4], a PU re-transmission based error control scheme is studied for the sake of designing an optimal transmission policy for the SU. Based on the PU re-transmission packet state, the SU determines its transmission strategy.

Seddik *et al.* presented in [5] a secondary access scheme that exploits the PU automatic repeat request (ARQ) feedback; the SU refrains from accessing the channel upon it hears a NACK feedback (FB) from the primary receiver (to allow for collision-free re-transmission, thus avoiding sure collisions). The scheme is shown to achieve higher secondary user throughput while guaranteeing the PU QoS constraint. Furthermore, in [6], besides soft-energy sensing, the authors propose the use of PU feedback information for designing the SUs' access scheme under a PU stability constraint. Afterward, the work in [7] introduced a partially observable Markov decision process (POMDP) framework to design a SUs' MAC protocol exploiting the history of the PU feedback bits. Intuitively, A greedy algorithm was proposed to simplify the solution for the proposed POMDP problem. The greedy algorithm's goal was to maximize the instantaneous SU reward. However, the work assumed perfect knowledge of the PU arrival rate, which might not be available in many real-life scenarios.

Moreover, there has been another direction for designing SUs' access schemes by exploiting the channel quality indicator (CQI) feedback [8]. Besides, the work in [9] has examined the design of hybrid schemes that exploit both ARQ and CQI feedback. However, all of the above works have some assumptions (like the knowledge of the PU arrival rate to optimize the SUs' access decisions), which might not be practical scenarios.

In this paper, we focus on the design of SUs' access schemes exploiting the primary network available feedback information in the form of ARQ and CQI feedback bits. The problem of the design of the SUs' access scheme is formulated as a POMDP. We focus on the use of different reinforcement learning (RL) approaches, namely Q-Learning, Deep Q-Network (DQN), and Deep Deterministic Policy Gradient (DDPG), to design SUs' access schemes (i.e., solve the formulated POMDP problem) that require minimal knowledge about the PU parameters. The proposed RL based schemes are shown to achieve the same performance of the previously proposed schemes (which have some impractical, or hard to achieve, assumptions like knowing the PUs' arrival rates). Our proposed schemes would allow for online implementations that can adapt to variations in the primary network, which is not the case with the previously proposed "static" algorithms.

II. System Model

A time-slotted system is considered in our model, which consists of one PU and one SU for the ease of exposition [1]. The PU hosts an infinite buffer to store incoming packets. The PU average arrival rate is λ_p packets per time slot; the arrival process is assumed to be a Bernoulli process with independent and identically distributed arrivals. The packet's transmission time is assumed to fit exactly within the duration of one slot. Therefore, λ_p will assume values in the range $0 \leq \lambda_p \leq 1$. Otherwise, the stability of the PU queue will not be attained. Also, it is assumed that the SU queue is always backlogged, i.e., the SU will always have a packet to transmit.

The SU employs a random access scheme with access probability $a_s(.)$, i.e., a slotted ALOHA access scheme is adopted. The access probability will be adapted depending on the SU estimate of the PU activity (PU state). In our proposed model, and as will be discussed later, the access probability will be a function of the PU feedback, whether ARQ and/or CQI feedback states.

Finally, we assume a collision channel model. If either the PU or the SU transmits in any time slot, this will result in a successful transmission. Packets can only be lost in transmission in the case of concurrent PU and SU transmissions. In this case, a collision is declared, and all the packets involved will be lost. The PU receiver will send a NACK feedback for the PU to re-transmit its packet. We also assume that all feedback information is always received correctly at the receiving nodes as these feedback bits are normally well protected by strong channel codes.

A. POMDP Framework

We model the system dynamics using a POMDP model, as explained next. RL algorithms are used to learn policies to solve the proposed POMDP. For ease of exposition, and due to space limitations, we will limit our attention in this paper to the POMDP model for the system exploiting the ARQ feedback. The model can be readily extended to the cases of CQI and hybrid (ARQ-CQI) systems.

In the ARQ based system, we have three possible PU ARQ feedback states: ACKs, NACKs, and No-Feedback (No-FB). An ACK denotes a successful transmission, a NACK denotes a failed transmission, and No-FB means that the PU did not attempt any transmission in the last time slot. In an ideal, perfect-sensing system, the SU will have perfect knowledge of the PU state (i.e., active or idle) in each time slot. On the contrary, in our system, we assume that the SU has access to the PU feedback information only. Therefore, a POMDP model fits well to our problem, since the SU has partial information about the PU activity. As a consequence, a belief vector is constructed at the SU which represents the Markov chain states of the PU queue.

As shown in Fig. 1, two classes of states are defined, which are the i_F's and i_R's states. The subscript F refers to the first transmission, while the subscript R indicates a PU re-transmission state, after receiving a NACK feedback.

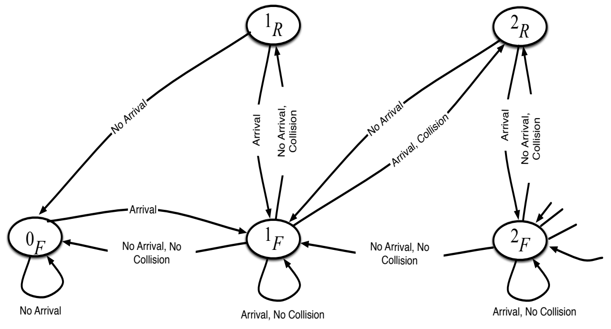

Fig. 1: The PU queue Markov chain model

A POMDP is defined by the tuples $(\mathcal{S},\ \mathcal{A},\ \mathcal{O},\ T,\ \Omega,\ R)$, where the set $\mathcal{A}$ denotes the set of SU actions (which correspond to different SU access probabilities). The set $\mathcal{S}$ denotes the PU Markov chain states $S = \{\{i_F\}, \{j_R\}\}$, $i = 0, 1, \cdots$ and $j = 1, 2, \cdots$. The set $\mathcal{O}$ defines the observations set that is presented by $\mathcal{O} = \{\text{ACK, NACK, No-FB}\}$.

The function $T(.)$ denotes the transition probabilities function, where $T(s'|s, a)$ indicates the likelihood to go from state s to state s' given action a. To better illustrate it, we give few examples below for $T(s'|s, a)$, for different values of s, s' and a.

$$\begin{aligned}
&T(i_R|j_F,\ \text{no access}) = 0,\ \forall i,\ j\\
&T(i_F|j_F,\ \text{access}) \quad = 0,\ \forall i,\ j \neq 0\\
&T(1_F|0_F,\ \text{access}) \quad = \lambda_p,\\
&T(1_F|0_F,\ \text{no access}) \quad = \lambda_p,\\
&T(i_R|j_F,\ \text{access}) \quad = \begin{cases} 1-\lambda_p & \text{if } i=j,\ j\neq 0\\ \lambda_p & \text{if } i=j+1,\ j\neq 0\\ 0 & \text{otherwise,}\end{cases}\\
&T(i_F|j_R,\ \text{no access}) = \begin{cases} 1-\lambda_p & \text{if } i=j-1\\ \lambda_p & \text{if } i=j\\ 0 & \text{otherwise.}\end{cases}
\end{aligned} \tag{1}$$

The function $\Omega(o|s', a)$ denotes the probability of observing o given that action a is applied to result in state s'. It can be estimated as [2]

$$\Omega(o|s' = i_F,\ \text{no access})$$
$$= \begin{cases} 0 & o = \text{NACK and } \forall i_F\\ P_{\text{ACK}}(0_F) & o = \text{ACK},\ i_F = 0\\ P_{\text{No-FB}}(0_F) = 1 - P_{\text{ACK}}(0_F) & o = \text{No-FB},\ i_F = 0\\ P_{\text{ACK}}(1_F) & o = \text{ACK},\ i_F = 1\\ P_{\text{No-FB}}(1_F) = 1 - P_{\text{ACK}}(1_F) & o = \text{No-FB},\ i_F = 1\\ 1 & o = \text{ACK},\ i_F \geq 2\\ 0 & o = \text{No-FB},\ i_F \geq 2\end{cases} . \tag{2}$$

[1]Extending the model to the case of multiple PUs and SUs, under TDMA in the primary network is straightforward.

[2]By abuse of notations, we set $\Pr(A|B) = 0$ if $\Pr(B) = 0$ (for example we set $\Omega(o|i_R,\ \text{no access}) = 0$ since $\Pr(i_R,\ \text{no access}) = 0$ under our collision system model assumption).

$$\Omega(o|i_F,\ \text{access}) = \begin{cases} 0 & \forall o \text{ and } i_F \geq 2 \\ 1 & o = \text{No-FB},\ i_F = 0,1 \\ 0 & o = \text{ACK or NACK},\ i_F = 0,1 \end{cases} \tag{3}$$

$$\Omega(o|i_R,\ \text{no access}) = 0 \quad \forall o \tag{4}$$

$$\Omega(o|i_R, \text{access}) = \begin{cases} 1 & o = \text{NACK} \\ 0 & \text{otherwise} \end{cases}. \tag{5}$$

The function $R(.)$ denotes the reward function calculated as follows.

$$R(s,a) = \begin{cases} w & a = \text{access},\ s = 0_F \\ -1 & a = \text{no access},\ \forall s = 0_F \\ 1 & a = \text{no access},\ \forall s \neq 0_F \\ -1 & a = \text{access},\ s \neq 0_F \end{cases}. \tag{6}$$

It is an immediate reward that the SU has earned for taking a specific action and reaching a new state. If the queue of the PU is empty, i.e., $s = 0_F$ and the SU accessed the channel, it will gain a positive reward. However, in this same case and if the SU does not access the channel, it receives a penalty due to the lost transmission opportunity. Moreover, if the queue is not empty and the SU accessed, it also receives a penalty. On the other hand, if the SU does not access the channel, it receives a positive reward for avoiding a sure collision with the PU.

The belief vector is given by $\mathbf{b}(s_t) = [b(0_F)_t,\ b(1_F)_t,\ b(1_R)_t,\ \cdots]$, where t is the time index. After taking an action a_t and observing some o_{t+1}, the new belief for some state s_{t+1} at time $(t+1)$ is given by

$$b(s_{t+1}) = \eta\Omega(o_{t+1}|s_{t+1}, a_t) \sum_{s_t \in \mathcal{S}} T(s_{t+1}|s_t, a_t) b(s_t), \tag{7}$$

where η is a probability normalization factor given by

$$\eta = \frac{1}{\sum_{s_{t+1} \in \mathcal{S}} \Omega(o_{t+1}|s_{t+1}, a_t) \sum_{s_t \in \mathcal{S}} T(s_{t+1}|s_t, a_t) b(s_t)}.$$

For the system using CQI feedback values, the SU is assumed to have access to the PU CQI feedback. We assume to have a binary CQI feedback in the form of "GOOD" or "BAD" channel. Upon receiving a good channel CQI feedback, the PU accesses the channel to transmit the packet on the top of its data queue. Upon receiving a bad channel CQI feedback, the PU refrains from accessing the channel as the PU transmission will certainly fail in this case. Again, allowing the SU to have access to the PU CQI would allow the SU to have some extra information to have better insight into the PU activity.

The PU channel is modeled as a two-state Markov chain with "GOOD" and "BAD" states, as mentioned above. Let p_G denotes the probability of the channel being in the Good state, and p_B denotes the probability of the channel being in the bad state. Let ζ_G and ζ_B be the steady-state probabilities of the channel being in the good and bad states, respectively. They can be easily shown to be given by

$$\zeta_G = \frac{1 - p_B}{2 - p_B - p_G}, \quad \text{and} \quad \zeta_B = \frac{1 - p_G}{2 - p_B - p_G}.$$

Finally, in the hybrid ARQ-CQI system, we assume that the SU will have access to, both, the PU ARQ and CQI feedback. In all of the above systems, the SU access decision can be modeled as a POMDP. Allowing the SU to access the PU feedback enables the SU to have "partial knowledge" about the PU activity and this should result in better SU access decisions.

It should also be noted that solving the formulated POMDP, using an RL-based approach, will always guarantee the stability of the PU queue. As in the case of an unstable PU queue, the SU will always collide with the PU packets whenever it attempts to transmit any packet. In this case, the reward function is minimized, not maximized, and this cannot be the solution resulting from our RL-based algorithms.

B. POMDP MAC Policy

This section describes the mapping of the belief vector to the action space. This mapping is affected by the reward of the current state as well as the expected reward in the following states, which is governed by the dynamics of the Markov chain. This is attributed to the fact that the belief vector in the next time instant will be affected by the present action. It is possible to model the MAC policy as a Markov decision process based on the belief vectors (belief MDP). Based on a belief vector $\mathbf{b}$ and an action a, the expected reward is given by

$$r(\mathbf{b}, a) = \sum_{s \in \mathcal{S}} b(s) R(s, a). \tag{8}$$

For any belief vector $\mathbf{b}$, the SU access policy π is defined by an action $a_\pi = \pi(\mathbf{b})$[3]. Over an infinite horizon, the accumulated reward is the objective function to be maximized. Starting with a belief vector $\mathbf{b}_0$, the estimated reward for policy π is given by

$$J^\pi(\mathbf{b}_0) = \sum_{t=0}^{\infty} \gamma^t r(\mathbf{b}_t, a_t) = \sum_{t=0}^{\infty} \gamma^t E\Big[R(s_t, a_t) \mid \mathbf{b}_0, \pi\Big] \tag{9}$$

where $0 \leq \gamma < 1$ is a discount factor. The optimal policy π^* is given by

$$\pi^* = \underset{\pi}{\text{argmax}} \ \ J^\pi(\mathbf{b}_0) \tag{10}$$

where $\mathbf{b}_0$ is the initial belief vector as defined above.

For each belief state, the maximum expected reward value specifies the optimal policy, π^*. It is closely modeled by the best value function V^*, which is the solution for the following Bellman equation

$$V^*(\mathbf{b}) = \max_{a \in \mathcal{A}} \Big[r(\mathbf{b}, a) + \gamma \sum_{o \in \mathcal{O}} \Omega(o \mid \mathbf{b}, a) V^*(\tau(\mathbf{b}, a, o))\Big], \tag{11}$$

[3]Note the optimal policy can be defined as a probability measure over the action space that is a function of the belief vector, i.e., the policy defines the probability for each action under a certain belief vector.

where $\tau(\cdot,\cdot,\cdot)$ is the belief state transition function.

After introducing our POMDP model and the MAC design policy, it should be noted that in a real-life scenario we might not be able to construct our belief vectors $\mathbf{b}$ based on the transition probability function $T(.)$. For example, if the SU does not know λ_p then it will not be able to construct the belief vector, $\mathbf{b}$, of the PU. Therefore, and unlike the work in [7], we propose to implement an RL based MAC that can efficiently learn in a model-free systems in which the underlying dynamics are not fully characterized. Next, we will present different RL algorithms that can be employed to design our SU MAC scheme and compare their performance later via extensive simulations.

III. ϵ-Greedy Q-Learning Algorithm

Q-Learning is one of the most common approaches in reinforcement learning. As in [10], at each time step, the SU in state s_t chooses an action a_t and goes to the next state s_{t+1} while receiving reward r_t. Through computing and collecting the experiences s_t, a_t, r_t, and s_{t+1}, the RL agent (SU in our case) can compute the state-action value function $Q(s,a)$, which is the expected overall future discounted reward when the SU takes an action a in state s. The update equation for the Q-values is given by:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t)+\alpha\left[r_t+\gamma \underset{a}{max} Q(s_{t+1},a)-Q(s_t,a_t)\right]. \quad (12)$$

In the given formula, two major parameters that influence the equation are α and γ. The parameter α is the learning rate where $0<\alpha\leq 1$. It affects the extent to which the Q-values are changed by following an action. On the other hand, γ denotes the discount factor. It takes values in the range from zero to one and controls the influence of future rewards on the Q-value. The Q-Learning algorithm builds a lookup table that lists a Q-value for each state-action pair. The SU optimal policy is the deterministic policy that selects the action that has the highest Q-value in each state. Therefore, the optimal policy, $\pi^*(s)$, is given by

$$\pi^*(s) =_a Q(s,a).$$

To learn the Q-values, we adopt the ϵ-greedy approach [11]. Accordingly, the agent decides between exploiting the best-known action so far (i.e., the one that results in the highest Q-value) and exploring new actions that might result in higher rewards. The parameter ϵ is used as the probability of exploring new actions. Normally, we start with a high ϵ to explore more, not to be trapped in a local minimum. The value of ϵ should decay with time.

A. Deep Q-Learning (DQN) Algorithm

Deep Q-Learning is a combination of Reinforcement Learning and Deep Learning. DQN emerged to solve infinite-state MDP problems by learning a parametric approximation to the $Q(s,a)$ function. Moreover, the ϵ-greedy policy is used with DQN. This gives more opportunity for random exploration of actions as our machine learns to approximate the Q-values. Through exploration, the SU will be able to explore a variety of actions in different states at different arrival rates.

By discretizing the action space (i.e., discretizing the values of the access probabilities), DQN can be used to learn the best action for each PU feedback state. We model the access decision as a classification neural network, and for each feedback state, the machine should output the best action from the set of discrete actions. The machine does this by estimating the Q-value for each action (i.e., the machine will have a number of outputs that equals the number of actions, each estimating the Q-value corresponding to a specific action). The best action at each state will be the action that results in the highest Q-value.

For forward and backward propagation, the expected Q-value for the state is called the target function. The Q-value related to the action taken by the agent is updated while the other actions' Q-values are kept untouched. The target function uses the Bellman equation to estimate the value function of the action as in (12). Therefore, we have the following update rule for the target Q-value at the $i+1$-th iteration

$$Q^{i+1}(s,a) = r + \gamma \underset{a'}{max}\, Q^i(s',a'), \quad (13)$$

where s represents the state, a corresponds to the action, r is the instantaneous reward, and s' is the next state. (13) estimates $Q(s,a)$ as the reward plus the maximum predicted Q-value of the following state discounted by γ, which is a discount factor as defined before. Eventually, this value is supposed to be predicted and learned by the model for every pair (s,a). After estimating the action-value target function, the best policy is shown to be a greedy policy, i.e., in each state s, the policy selects the action a as $\pi^*(s)=\arg\max_a Q^*(s,a)$.

The loss function we adopt in our work is the Mean square error (MSE), which calculates the square of the difference between the predicted Q-value and the current estimated Q-value as follows [12]

$$L(\theta^{(i)}) = E\left[(y^{(i)} - Q(s,a;\theta^{(i)}))^2\right]$$

$$y^{(i)} = E\left[r+\gamma \underset{a'}{max}\, Q(s',a':\theta^{(i-1)})|s,a\right],$$

where $\theta^{(i)}$ represents the set of the neural network parameters at iteration i. Afterwards, back-propagation is done by applying a gradient descent based update to the parameters to minimize the loss function until reaching convergence.

B. Deep Deterministic Policy Gradient

As we will show later, both Q-Learning and DQN achieve better performance compared to conventional schemes. However, both still have their drawbacks; they do not entirely explore the full action space due to their discretization of the action space. Motivated by this, a model-free Deep Deterministic Policy Gradient (DDPG) algorithm is proposed [13]. As an actor-critic off-policy algorithm, it uses a Deep Neural Network (DNN) to learn and explore continuous access probabilities in the action space. The DDPG approach is divided into four networks, two networks for the actor and the critic, and two for the target actor and target critic. Through a parameterized actor function $\mu(s;\theta^{\mu})$, with parameter vector θ^{μ}, the actor maps states to actions that specify the current policy. On the other side, the critic $Q(s,a)$ maps states and the actor output (action) to a Q-value function using the Bellman equation.

The actor-network is updated by calculating the loss to the target actor-network, which is updated slowly to ensure its convergence. The actor update equation can be derived as

$$\begin{aligned}\nabla_{\theta^{\mu}} &\approx E[\nabla_{\theta^{\mu}} Q(s,a;\theta^{Q})|_{s=s^k,a=\mu(s^k|\theta^{\mu})}] \\ &\approx E\Big[\nabla_a Q\Big(s,a;\theta^{Q}\Big)|_{s=s^k,a=\mu(s^k)}\nabla_{\theta^{\mu}}(s;\theta^{\mu})|_{s=s^k}\Big],\end{aligned} \tag{14}$$

where θ^{Q} is the parameter of the critic network (used to predict the Q-values) and s^k is the current state. The actor-network aims to drive the policy that maximizes the long term reward defined as

$$\mu^* = argmax_{\mu} E_{\mu}\Big[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots + \gamma^{k-1} r_k\Big]. \tag{15}$$

IV. Numerical Results

In our simulations, and for the case of discrete action space (i.e., the case of Q-Learning and DQN), we assume 11 possible actions: $\mathcal{A}$ = {no access, access with probabilities $0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. We start with $\epsilon = 1$ to favor exploration as the machine begins to learn, not to be trapped in a local minimum. We set a decaying ϵ; the decaying factor is 0.999, with a min exploration rate of 0.01. The learning rate is set to be $\alpha = 0.0001$; although it is very small and the machine takes a longer time to converge, it resulted in a stable set of actions each time the model is trained based on our trials. The discount factor is set to be $\gamma = 0.95$, which is the discount factor for future rewards. It should be noted that the selection of our hyperparameters (like the learning rate, discount factor, etc) is based on many trials and we have selected the set of hyperparameters that resulted in the best SU throughput results.

First, we investigate the effect of the choice of the reward function on our system performance. We compare five different values for r as shown in Fig. 2 for the Q-Learning based approach. Higher values of r resulted in a slightly degraded performance, since they caused the SU to aggressively access the channel which resulted in collisions with the PU especially at higher PU arrival rates. The best performance was achieved by the algorithm with $r = 1, 2, 10, 50$ with a slight difference in the resulting access probabilities (actions) for different arrival rates.

Fig. 3 depicts a performance comparison, in terms of the SU throughput, between the Q-Learning algorithm and the greedy algorithm proposed in [7] as well as the FB-based approach proposed in [5]. We assume that collisions between the PU and the SU are the only source of error. In the greedy approach of [7], $w = 0$ means that the SU will always attempt transmission in the case of ACK or no feedback. As a consequence, it

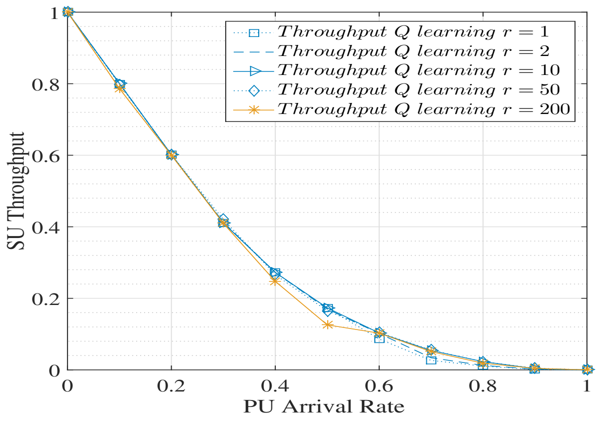

Fig. 2: ARQ feedback using RL (Q-Learning) with different rewards

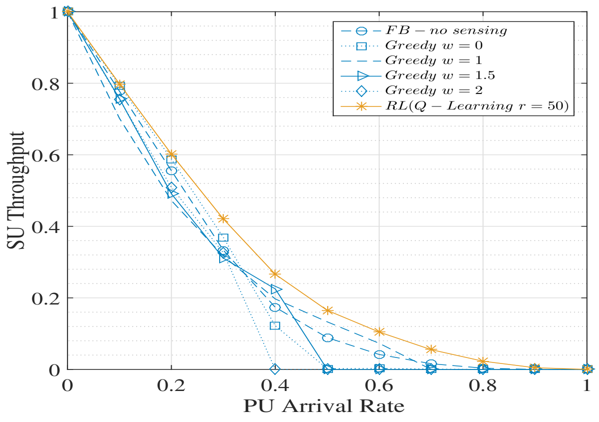

Fig. 3: ARQ feedback-based access using RL (Q-Learning) with $r = 50$ versus the greedy algorithm [7] and the FB-based approach of [5]

degrades the SU throughput at high PU arrival rates due to frequent collisions. As w increases, there is less tendency for the SU to access the channel. Hence, losing more transmission opportunities over the channel for low PU arrival rates.

As noticed from fig. 3, the best performance corresponds to the case of RL with $r = 50$. It causes the SU to aggressively access the channel compared to the cases with a lower r. For example, at an arrival rate of 0.6, the access probability is 1 for the cases of no feedback and ACK, while with $r = 1$, the access probabilities are 0.4 for the no feedback and 0.2 for the ACK. For the case of $r = 50$, the access probability for the NACK observations is always zero for all arrival rates as the SU refrains from accessing the channel in this case, giving a chance for the PU to transmit its packet collision-free.

Next, we consider the deep Q-Learning network (DQN) approach. The deep Q-Learning network architecture consists of three layers: the input layer, one hidden layer, and the output layer. First, the network input layer is composed of three nodes with a one-hot encoded state vector (3×1). It represents the

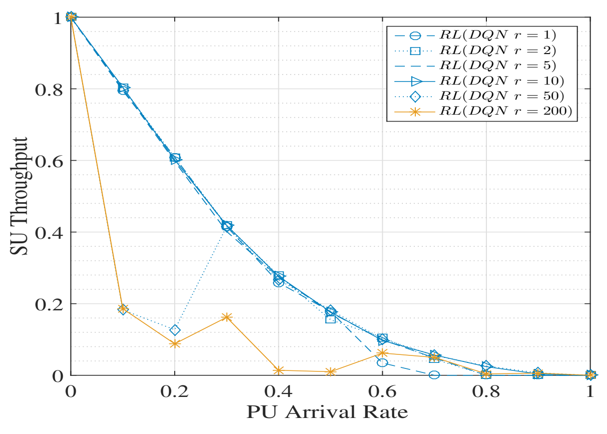

Fig. 4: ARQ feedback using RL (DQN) with different rewards

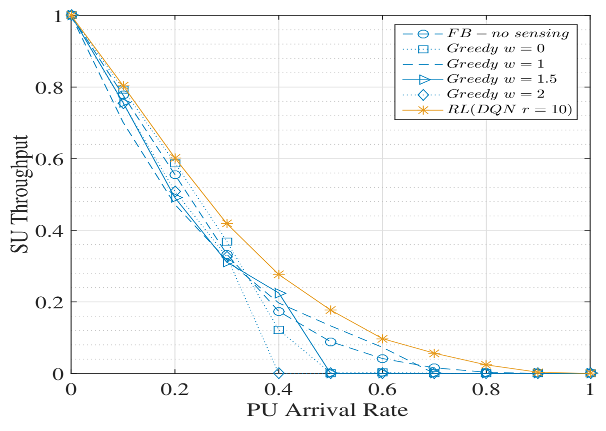

Fig. 5: ARQ feedback using RL (DQN with $r = 10$) vs conventional methods

three observations (ACK, NACK, and no feedback). Second, we have a hidden layer with ten fully connected neurons (10×1) with a Sigmoid activation function. Finally, the output layer consists of eleven nodes that correspond to the eleven actions of the SU (11×1) with a linear activation function. The optimizer used in the prediction is the Adaptive Moment Estimation (ADAM) [14].

Fig. 4 shows the effect of the reward function value on the performance of the DQN-based approach. Similar to the case of Q-Learning, the value of r presents a trade-off between aggressively accessing the channel for higher r values and missing transmission opportunities for smaller r values. From fig. 4, we can see that a value of $r = 10$ results in the best SU performance. Moreover, fig.5 compares the performance of the DQN approach for $r = 10$ to the greedy algorithm for different w values [7] and also to the FB-based approach of [5]. Clearly, the DQN-based approach outperforms the two access scheme while assuming minimal information about the primary user (e.g., it does not assume any knowledge of the PU arrival rate while the other approaches assume perfect knowledge of the PU arrival rate).

Next, we compare Q-Learning to DQN in fig.6, which shows similar performance, because DQN is designed for large input state space that can not be totally visited by Q-learning. However, in our case, we have small input state [12]. In terms of convergence time, DQN takes longer to converge compared to Q-Learning, which gives an advantage for Q-Learning.

Next, we consider the DDPG-based approach. The actor-critic network architecture consists of four layers: one input layer, two hidden fully connected layers of 64 neurons each, and one output layer. ADAM is used to learn the parameters of the neural network with a learning rate of 10^{-4} and 10^{-3}, respectively. A non-linear rectifier is used in all hidden layers beside a tanh function in the output layer for bounding the actions. The mini-batch size is set to be 64, and the replay buffer size is set to be 10^6. For exploration, Ornstein-Uhlenbeck noise was added to the action output. [15]

Fig. 7 compares the performance of the CQI-based access

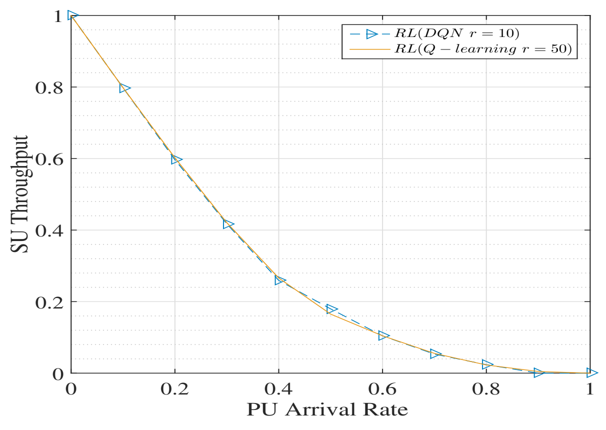

Fig. 6: ARQ feedback using RL (DQN vs Q-Learning algorithm)

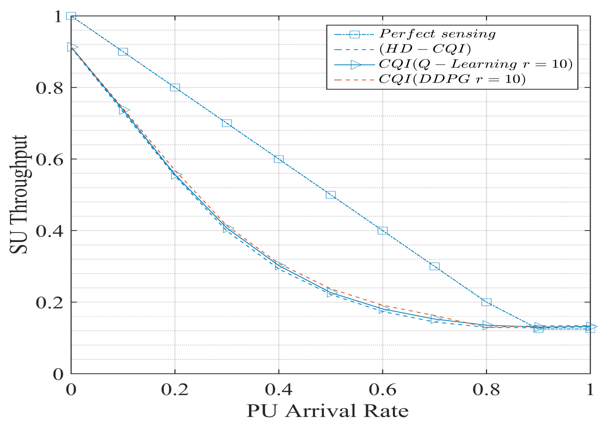

Fig. 7: CQI feedback with RL (Q-Learning, DDPG) vs HD-CQI [8]

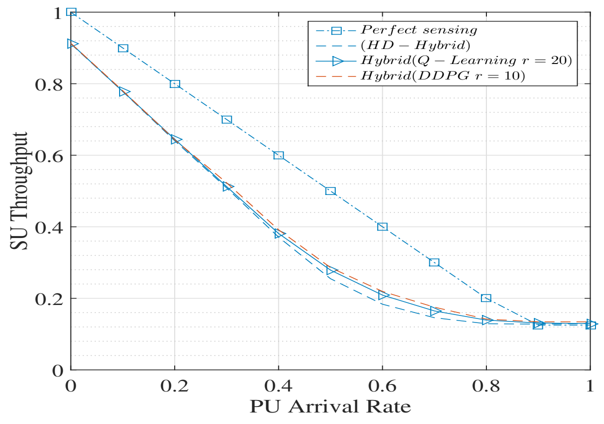

Fig. 8: Hybrid feedback with RL (Q-Learning, DDPG) vs HD-Hybrid [9]

schemes of the two proposed RL algorithms, namely, Q-Learning and DDPG, to the baseline approach of [8] (where a closed-form expression is derived) and the perfect sensing upper bound. The probability of false alarm p_F is set to 0.1 and the probability of detection p_D is set to 0.9. Moreover, the probability of the channel staying in the good p_G, and the bad p_B are 0.9 and 0.3, respectively. From this figure, it is clear that the DDPG approach yields the best performance with a slight performance gain over the Q-Learning approach. This is attributed to the fact that DDPG exploits the whole space of action probabilities and it does not discretize the action space. The conventional method in [8] results in the worst performance. It should be noted that at zero PU arrival rate, the throughput is limited by the false alarm probability as the PU queue is always empty in this case, which is $1 - p_F = 0.9$. Also, at high arrival rates, the throughput converges to the steady-state probability of the channel being in the bad state, which is 0.125 in our case.

Finally, Fig 8 compares the SU throughput for four access schemes, namely, the conventional (HD-Hybrid) approach [9], the proposed Q-Learning and DDPG approaches, in addition to the perfect sensing upper bound. Again, and as mentioned above, at zero PU arrival rate, the throughput is limited by the false alarm. We use the same simulation parameters of Fig 7. Clearly, the DDPG-based approach yields the best performance, achieving a slight throughput gain over Q-learning. For example, at a PU arrival rate of 0.6, the SU access probability learned by Q-Learning in the ACK and good-CQI feedback is 0.5 achieving a throughput of 0.209. On the other hand, with DDPG, the learned access probability is 0.44, which achieves a slightly better SU throughput of 0.219. Allowing continuous action space in the case of DDPG can, in general, result in attaining higher rewards.

V. Conclusion

In this paper, we consider the problem of designing SUs' access scheme in a cognitive radio system exploiting the available PUs feedback information in the form of ARQ and CQI feedback. We consider three systems; one in which the SU has access only to the ARQ feedback, one in which the SU has access only to the CQI feedback, and one in which the SU has access to both the ARQ and CQI feedback. The problem of SUs access decision is modeled as a Partially Observable Markov Decision Process (POMDP) which is solved using different reinforcement learning (RL) based approaches, namely, Q-Learning, DQN and DDPG. Contrary to prior work, all of the RL based approaches assume minimal knowledge about the PU in terms of PU arrival rate. Our results show the merits of the proposed RL approaches in learning the PU activity based only on observing its feedback. Finally, it should be noted that the proposed approaches in this paper can be used on top of any PU sensing scheme (e.g., energy sensing) and they are always guaranteed to result in a performance gain.

References

[1] I. Akyildiz, W. Lee, M. Vuran, and S. Mohanty, "Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey," *Computer Networks*, vol. 50, no. 13, pp. 2127–2159, Sep. 2006.

[2] K. Eswaran, M. Gastpar, and K. Ramchandran, "Bits through arqs: Spectrum sharing with a primary packet system," in *2007 IEEE International Symposium on Information Theory*, June 2007, pp. 2171–2175.

[3] S. Huang, X. Liu, and Z. Ding, "Distributed power control for cognitive user access based on primary link control feedback," in *IEEE International Conference on Computer Communications (INFOCOM)*, San Diego, CA, March 2010.

[4] M. Levorato, U. Mitra, and M. Zorzi, "Cognitive interference management in retransmission-based wireless networks," *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3023–3046, 2012.

[5] K. Seddik, A. Sultan, A. El-Sherif, and A. Arafa, "A feedback-based access scheme for cognitive radio systems," in *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, San Francisco, CA, June 2011.

[6] A. Arafa, K. Seddik, A. Sultan, T. ElBatt, and A. El-Sherif, "A feedback-soft sensing-based access scheme for cognitive radio networks," *Wireless Communications, IEEE Transactions on*, vol. 12, no. 7, pp. 3226–3237, 2013.

[7] K. G. Seddik and A. A. El-Sherif, "A pomdp framework for cognitive mac based on primary feedback exploitation," in *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2014, pp. 1281–1285.

[8] S. A. Attalla, K. G. Seddik, A. A. El-Sherif, and S. I. Rabia, "Soft-sensing cqi feedback-based access scheme in cognitive radio networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 486–499, 2018.

[9] ——, "Hybrid arq-cqi feedback-based access scheme in cognitive radio networks," *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2019.

[10] J. Schilperoort, I. Mak, M. Drugan, and M. Wiering, "Learning to play pac-xon with q-learning and two double q-learning variants," in *Learning to Play Pac-Xon with Q-Learning and Two Double Q-Learning Variants*, 09 2018.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[12] J. Liu, B. Krishnamachari, S. Zhou, and Z. Niu, "Deepnap: Data-driven base station sleeping operations through deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4273–4282, 2018.

[13] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[15] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, 09 2015.